

# Metrics for the Case Management Modeling and Notation (CMMN) Specification

Mike A. Marin  
University of South Africa  
IBM Analytics Group  
25131 Mackenzie Street  
Laguna Hills, CA 92653, USA  
mmarin@acm.org

Hugo Lotriet  
College of Science  
Engineering and Technology  
University of South Africa  
Florida Park, Johannesburg,  
South Africa  
lotrihh@unisa.ac.za

John A. Van Der Poll  
Graduate School of Business  
Leadership (SBL)  
University of South Africa  
Midrand, 1686, Gauteng,  
South Africa  
vdpolja@unisa.ac.za

## ABSTRACT

The Case Management Modeling and Notation (CMMN) specification, published by the Object Management Group (OMG) in 2014, describes a declarative style for modeling business processes. The declarative nature of CMMN is intended to supplement the procedural style of the Business Process Modeling and Notation (BPMN). Although multiple metrics have been developed and verified for BPMN, the authors are not aware of any metrics developed for CMMN. Being a relative new process specification the understanding of complexity metrics for CMMN ought to be beneficial for practitioners and researchers by providing a way to compare case management models.

This study provides a formal description of CMMN and three metrics are defined, namely size, length, and complexity. The metrics are theoretically validated using the formal framework for software measurements defined by Briand *et al.* and the complexity metric is further validated using Weyuker's properties for software complexity measures.

## CCS Concepts

•General and reference → Metrics; •Applied computing → Business process modeling; •Software and its engineering → System modeling languages; •Computing methodologies → Model verification and validation;

## Keywords

Case management, Case handling, CMMN, BPMN, Modeling complexity, Complexity metrics, Process modeling complexity

## 1. INTRODUCTION

This research defines three metrics for the Case Management Modeling Notation (CMMN) [25]. The proposed metrics are the size  $CS$ , the length  $CL$ , and complexity  $CC$  of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

SAICSIT '15, September 28-30, 2015, Stellenbosch, South Africa

© 2015 ACM. ISBN 978-1-4503-3683-3/15/09...\$15.00

DOI: <http://dx.doi.org/10.1145/2815782.2815813>

a CMMN model.

Most authors are in agreement that software metrics should be theoretically and empirically validated [22, 23, 30]. The metrics proposed in this study are theoretically validated using an axiomatic approach and future work will focus on the empirical validation. The Briand *et al.* framework [4] and Weyuker properties [36] are categorized as axiomatic or property-based approaches to validation [30], and they are commonly used for validating software metrics [23].

Business process improvement practitioners are increasingly interested in case management in many industries [14]. However, it is known that users find complex process models difficult to understand, and complex models are more likely to contain design errors [18]. Therefore, research on complexity metrics of process models is important, and several studies on the subject have been conducted [1, 5, 6, 11, 12, 31, 35, 18, 21, 26]. In those studies the researchers have used UML, BPMN, workflow nets, or proprietary BPM product models, all of which are based on directed graphs and are procedural in nature. CMMN defines a declarative style that is different to the BPM procedural style [19]. Therefore, the metrics defined for procedural models may not be applicable to CMMN. This study addresses that gap by proposing three new metrics for CMMN.

Section 2 provides brief background information about case management and the CMMN notation. Section 3 describes the methodology used, introduces a formal definition for CMMN models, and describes the three proposed metrics. Section 4 presents the theoretical validation for the proposed metrics using Briand *et al.* framework [4] and Weyuker properties [36]. Section 5 presents our findings and provides suggestions for future research. Finally, section 6 describes our conclusions.

## 2. BACKGROUND

Case management in the context of process technology was first introduced by Berkley and Eccles [2] in 1991 and Davenport and Nohria [8] in 1994. Case handling was introduced by Van Der Aalst and Berens in 2001 [33] and Reijers *et al.* [27] in 2003 to support the flexibility required by knowledge workers during a process and to help them better deal with exceptions that may occur during such process.

In 2009, the Object Management Group (OMG) issued a request for proposal (RFP) for the creation of a standard modeling notation for case management [24] to serve as a complement to its BPMN specification. The result was

CMMN [25], which was first published in 2014. The main difference between CMMN and BPMN is the shift from procedural to declarative models [19]. CMMN addresses case management as described in [33, 34, 27, 32] with a data-centric approach based on business artifacts [19]. Case management as defined by CMMN provides flexibility to knowledge workers with regards to what tasks or activities should be performed and when they should be performed [15].

Researchers are starting to look at CMMN and its applicability to case management requirements. Schönig *et al.* [28] in 2013 considered human centric processes starting with CMMN modeling skeletons that evolve over time. Marin *et al.* [20] in 2014 compared the CMMN method against other modeling notations. That study concluded that CMMN compares favorable to other process modeling notations like BPMN. Hauder *et al.* [13] in 2015 explored the applicability of CMMN for knowledge intensive processes exposed via a wiki environment for business users. The conclusion was that a wiki environment allows for non-technical users to structure knowledge intensive processes. Kurz *et al.* [17] in 2015 compared CMMN against adaptive case management and concluded that for the most part CMMN fulfills the adaptive case management requirements.

### 3. METHODOLOGY

This section starts by formalizing a CMMN model. We abstract and formalize the characteristics of a model that will allow for the definition of concrete metrics. The formalization will allow us to validate the metrics against formal frameworks like those of Briand *et al.* [4] and Weyuker [36].

Our focus is on complexity metrics for CMMN, and size and length are considered complexity metrics by Muketha *et al.* [23]. Therefore, we define three metrics, size  $CS$ , length  $CL$ , and complexity  $CC$  of a CMMN model. The metrics are inspired by Briand *et al.* [4] definitions of size, length, and complexity. Briand *et al.* assume a system with procedural characteristics, and uses directed acyclic graphs to describe its framework. That imposed some constraints while defining the metrics, because CMMN is declarative [19] instead of procedural.

The metrics are validated using the formal framework for software measurements defined by Briand *et al.* [4], and the complexity metric is further validated using the nine properties for complexity measures defined by Weyuker [36].

#### 3.1 Case Management Modeling and Notation (CMMN)

DEFINITION 1. (*Model*): A CMMN [25] model  $\mathcal{C}$  is a collection of model elements  $\mathcal{E}$  with annotators  $\mathcal{A}$  and related by scope  $\mathcal{U}$  and event  $\mathcal{V}$  relationships. A model is defined as a tuple

$$\mathcal{C} = \langle \mathcal{E}, \mathcal{U}, \mathcal{V}, \mathcal{A} \rangle \quad (3.1)$$

Where

$\mathcal{E}$  is a set of modeling elements.  $\mathcal{E}$  is strongly typed and each element on  $\mathcal{E}$  belongs to one of the following types: scope, data, plan, or optional. Table 1 shows the CMMN elements.

$\mathcal{U}$  is a binary relationship in which two elements  $x$  and  $y$  in  $\mathcal{E}$  are related if and only if they are contained in the same scope. Note that  $\llbracket x, y \rrbracket \in \mathcal{U}$  is an unordered pair.


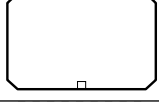
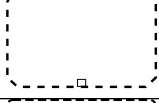
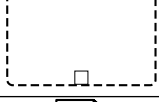




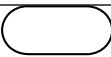
type	Element ( $\mathcal{E}$ )	Name
Scope		case
		stage
		discretionary stage
		plan fragment
Data		case file item
Plan		task
		discretionary task
		event listener
		milestone
Optional	.....	connector (sentry)

Table 1: CMMN elements

$\mathcal{V}$  is a binary relationship in which two elements  $x$  and  $y$  in  $\mathcal{E}$  are related if and only if an event from one ( $x$ ) triggers the other ( $y$ ). Note that  $\langle x, y \rangle \in \mathcal{V}$  is an ordered pair.

$\mathcal{A}$  is a set of annotators used to indicate characteristics of elements in  $\mathcal{E}$ . Each annotator ‘a’ in  $\mathcal{A}$  is related to one and only one element  $x$  in  $\mathcal{E}$ . There are three types of annotators, namely decorators, sentries, and markers. Most elements  $x$  in  $\mathcal{E}$  can be associated with a single marker, one of each decorator (collapsed or expanded), and multiple sentries. Table 2 shows the annotators.

DEFINITION 2. (*Scope element set*): A scope element is an element that can contain other elements. The set of scope elements  $\mathcal{M}$  is a subset of  $\mathcal{E}$  ( $\mathcal{M} \subseteq \mathcal{E}$ ).

$$\mathcal{M} = \{z \mid \text{type-of}(z) \in \text{Scope}\} \quad (3.2)$$

Where,  $\text{type-of}(z)$  is described by type in Table 1.

As shown in Table 1, the scope elements are the case, stage, discretionary stage, and plan fragment.

DEFINITION 3. (*Case element set*): A case element is a special scope element  $z \in \mathcal{M}$  that starts a case definition. A model  $\mathcal{C}$  can contain multiple case definitions, each one with its corresponding case element. The set of case elements  $\mathcal{L}$  is a subset of scope elements ( $\mathcal{L} \subseteq \mathcal{M}$ ).

$$\mathcal{L} = \{z \mid \text{isa}(z) = \text{case}\} \quad (3.3)$$

Where,  $\text{isa}(z)$  is described by name in Table 1.

Any nonempty model  $\mathcal{C}$  must contain at least one case element. A case element can contain other elements, but it cannot be contained by any element.

Therefore,  $\mathcal{C} \neq \emptyset \iff \mathcal{L} \neq \emptyset$ .

DEFINITION 4. (*Module*): A module of a model  $\mathcal{C}$  is defined by a scope element  $z \in \mathcal{M}$ , as a tuple

$$\lceil z \rceil = \langle \mathcal{E}_z, \mathcal{U}_z, \mathcal{V}_z, \mathcal{A}_z \rangle \quad (3.4)$$

Where

$\mathcal{E}_z$  is a subset of  $\mathcal{E}$  defined as

$$\mathcal{E}_z = \{z\} \cup \{x \mid x \text{ is reachable from } z\}.$$

$\mathcal{U}_z$  is a subset of  $\mathcal{U}$  defined as

$$\mathcal{U}_z = \{\llbracket x, y \rrbracket \mid \llbracket x, y \rrbracket \in \mathcal{U} \wedge x \in \mathcal{E}_z \wedge y \in \mathcal{E}_z\}$$

$\mathcal{V}_z$  is a subset of  $\mathcal{V}$  defined as

$$\mathcal{V}_z = \{\langle x, y \rangle \mid \langle x, y \rangle \in \mathcal{V} \wedge x \in \mathcal{E}_z \wedge y \in \mathcal{E}_z\}$$

$\mathcal{A}_z$  is a subset of  $\mathcal{A}$  defined as

$$\mathcal{A}_z = \{a \mid a \in \mathcal{A} \wedge (\exists y)(y \in \mathcal{E}_z \wedge y = \text{annotated-by}(a))\}$$

Where,  $\text{annotated-by}(a)$  describes the element in  $\mathcal{E}$  that ‘ $a$ ’ annotates.

Note that  $z$  may contain other scope elements, in which case all the elements contained in these scope elements are also in  $\lceil z \rceil$ . In other words,  $x \in \mathcal{E}_z \wedge x \in \mathcal{M} \implies \mathcal{E}_x \subseteq \mathcal{E}_z$ . In addition, note that  $\mathcal{E}$  is partitioned by  $\mathcal{E}_x$  and  $\mathcal{E}_y$ ,

$$\mathcal{E} = \mathcal{E}_x \cup \mathcal{E}_y \wedge \mathcal{E}_x \cap \mathcal{E}_y = \emptyset \iff \mathcal{L} = \{x, y\} \quad (3.5)$$

By slight abuse of notation, we use  $\mathcal{E}_P$  with upper case  $P$  to indicate the set  $\mathcal{E}$  of model  $P$ , and  $\mathcal{E}_z$  with lower case  $z$  to indicate the set  $\mathcal{E}$  of module  $\lceil z \rceil$ .

DEFINITION 5. (*Scope relationship*): A scope relationship relates two elements if and only if they are in the same scope  $z \in \mathcal{M}$ . The scope is a binary relationship  $\mathcal{U}$  on  $\mathcal{E}$  ( $\mathcal{U} \subseteq \mathcal{E} \times \mathcal{E}$ ), defined as follows

$$\begin{aligned} \mathcal{U} = \{ \llbracket x, y \rrbracket \mid & x \neq y \wedge (\exists z)(z \in \mathcal{M} \wedge x \in \mathcal{E}_z \wedge y \in \mathcal{E}_z \\ & \wedge x \neq z \wedge y \neq z) \\ & \wedge \neg(\exists w)(w \in \mathcal{M}_z \wedge (x \in \mathcal{E}_w \vee y \in \mathcal{E}_w)) \} \end{aligned} \quad (3.6)$$

Due to the declarative nature of CMMN, there is no strong sequence or control flow relationship between its elements, but scope plays an important role in a CMMN model.

DEFINITION 6. (*Event relationship*): An event relationship relates two elements if and only if an event from one of them trigger an entry or exit criteria in the other element. The event is a binary relationship  $\mathcal{V}$  on  $\mathcal{E}$  ( $\mathcal{V} \subseteq \mathcal{E} \times \mathcal{E}$ ), defined as follows

$$\mathcal{V} = \{\langle x, y \rangle \mid \llbracket x, y \rrbracket \in \mathcal{U} \wedge x \xrightarrow{\text{event}} y\} \quad (3.7)$$

Events play an important role in CMMN, because entry and exit criteria for CMMN elements can be triggered by events. Events cannot cross scope, therefore the two modeling elements must already be related by scope.
















Type	Annotator ( $\mathcal{A}$ )	Name
Decorator		collapsed planing table
		expanded planing table
		auto complete
		collapsed
		expanded
		manual activation
		repetition
Sentry		required
		entry criterion
Marker		exit criterion
		non-blocking human
		process
		case
		participant
		timer

Table 2: CMMN Annotators

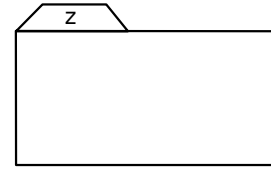


Figure 1: Model with one element (a case)

## 3.2 Examples

Three examples of models are presented. The example models will be used to illustrate CMMN concepts and grammar. The models are grammatically and semantically correct, but they are not intended to represent any real world process. Annotators are not labeled in CMMN, but for illustration purposes, dotted lines were used to associate annotator labels to the annotator icons in the model.

EXAMPLE 1. *Minimum case model*. Every model must contain at least one case element (see definition 3). Figure 1 shows the minimum valid CMMN model, with a single case element. The model has one model element  $\mathcal{E} = \{z\}$ , one scope element  $\mathcal{M} = \{z\}$ , and one case element  $\mathcal{L} = \{z\}$ . The scope relationship is empty  $\mathcal{U} = \emptyset$ . The event relationship is empty  $\mathcal{V} = \emptyset$ . The annotator set is empty  $\mathcal{A} = \emptyset$ . This example contains a single module  $\lceil z \rceil$  with  $\mathcal{E}_z = \{z\}$ . Note that  $\mathcal{E} = \mathcal{E}_z$ .

EXAMPLE 2. *Simple case model*. Figure 2 shows a model with a single case element  $w$  containing one case file item  $o$ , a connector  $p$ , and a task  $q$ . An event from case file item  $o$  triggers the entry criteria ‘ $k$ ’ that will allow task  $q$  to execute. The optional connector  $p$  visualizes the event propagation between case file item  $o$  and the entry criteria ‘ $k$ ’ of task  $q$ . The model has four model elements  $\mathcal{E} = \{w, o, p, q\}$ , one scope element  $\mathcal{M} = \{w\}$ , and one case element  $\mathcal{L} = \{w\}$ . The scope relationship is  $\mathcal{U} = \{\llbracket o, p \rrbracket, \llbracket o, q \rrbracket, \llbracket p, q \rrbracket\}$ . The event relationship is  $\mathcal{V} = \{\langle o, q \rangle\}$ . The annotator set is  $\mathcal{A} = \{k, j, l\}$ . This example contains one module  $\lceil w \rceil$  with  $\mathcal{E}_w = \{w, o, p, q\}$ . Note that  $\mathcal{E} = \mathcal{E}_w$ .

EXAMPLE 3. *Model with two cases*. Figure 3 shows a model with two case elements  $x$  and  $z$ . This model con-

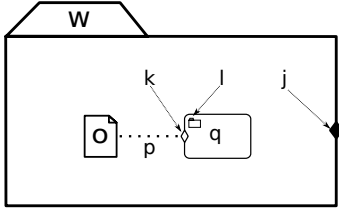


Figure 2: Model with four elements

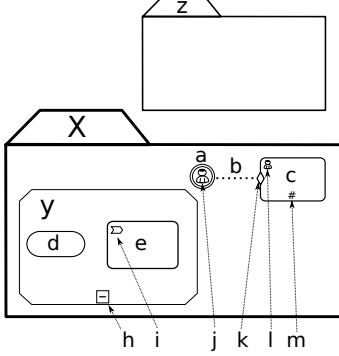


Figure 3: Model with eight elements

tains three scope elements  $\mathcal{M} = \{x, y, z\}$ , and two case elements  $\mathcal{L} = \{x, z\}$ . The model has eight modeling elements  $\mathcal{E} = \{a, b, c, d, e, x, y, z\}$ . There are three scopes,  $x$  with  $\{a, b, c, y\}$ ,  $y$  with  $\{d, e\}$ , and one empty scope  $z$ . The scope relationship  $\mathcal{U} = \{\llbracket a, c \rrbracket, \llbracket a, b \rrbracket, \llbracket a, y \rrbracket, \llbracket b, c \rrbracket, \llbracket b, y \rrbracket, \llbracket c, y \rrbracket, \llbracket d, e \rrbracket\}$ . Note that  $\{a, b, c\}$  and  $\{d, e\}$  are in different scope, and so,  $\llbracket a, d \rrbracket \notin \mathcal{U}$ . The same is true for  $\llbracket a, e \rrbracket, \llbracket b, d \rrbracket, \llbracket b, e \rrbracket, \llbracket c, d \rrbracket,$  and  $\llbracket c, e \rrbracket$  that are not in  $\mathcal{U}$ . The event relationship  $\mathcal{V} = \{\langle a, c \rangle\}$ . The annotator set  $\mathcal{A} = \{h, i, j, k, l, m\}$ . This example contains three modules,  $\lceil x \rceil$  with  $\mathcal{E}_x = \{x, a, b, c, y, d, e\}$ ,  $\lceil y \rceil$  with  $\mathcal{E}_y = \{y, d, e\}$ , and  $\lceil z \rceil$  with  $\mathcal{E}_z = \{z\}$ . Note that by (3.5)  $\mathcal{L} = \{x, z\} \implies \mathcal{E} = \mathcal{E}_x \cup \mathcal{E}_z$ . Figure 4 shows a tree view of the modeling elements  $\mathcal{E}$  in this example. The tree view is for illustration purposes and it is not a CMMN diagram.

### 3.3 Metrics

We define three metrics that are consistent with the formal framework for software measurements defined by Briand *et al.* [4]. Although we are interested in complexity metrics, we also define size and length, because Muketha *et al.* [23] concluded that size and length are similar to the complexity activity metrics proposed by Cardoso [6].

DEFINITION 7. (*Size metric*): The size of a model  $\mathcal{C}$  denoted by  $CS(\mathcal{C})$  is defined as the cardinality of  $\mathcal{E}$ ,

$$CS(\mathcal{C}) = |\mathcal{E}| \quad (3.8)$$

The size of a module  $\lceil z \rceil$  is defined as the cardinality of  $\mathcal{E}_z$ ,

$$CS(\lceil z \rceil) = |\mathcal{E}_z|$$

By (3.5) it follows that,

$$CS(\mathcal{C}) = |\mathcal{E}| = \sum_{z \in \mathcal{L}} |\mathcal{E}_z| = \sum_{z \in \mathcal{L}} CS(\lceil z \rceil)$$

Note that,

Size for Figure 1 is  $CS(\text{example1}) = |\mathcal{E}| = 1$ .  
 Size for Figure 2 is  $CS(\text{example2}) = |\mathcal{E}| = 4$ .  
 Size for Figure 3 is  $CS(\text{example3}) = |\mathcal{E}| = 8$ .  
 Size for  $\lceil x \rceil$  in Figure 3 is  $CS(\lceil x \rceil) = |\mathcal{E}_x| = 7$ .  
 Size for  $\lceil y \rceil$  in Figure 3 is  $CS(\lceil y \rceil) = |\mathcal{E}_y| = 3$ .  
 Size for  $\lceil z \rceil$  in Figure 3 is  $CS(\lceil z \rceil) = |\mathcal{E}_z| = 1$ .

DEFINITION 8. (*Length metric*): The length of a model  $\mathcal{C}$  denoted by  $CL(\mathcal{C})$  is defined as the maximum nesting depth of a model. The length  $CL(\mathcal{C})$  can be calculated by the following algorithm,

```

int funct CL(C) ≡
begin
  int m := 0
  for each case-plan c ∈ L do
    m := max(m, depth(m, c)) od
  return m
end.

int funct depth(s) ≡
begin
  int d := 0
  if (s ∉ M) then return 0 fi
  for all e in scope s do
    d := max(d, depth(e)) od
  return d + 1
end.

```

The length of a module  $\lceil z \rceil$  is defined as the maximum nesting depth of the module, and can be calculated using the following algorithm,

```

int funct CL(⟦z⟦) ≡
begin
  assert(z ∈ M)
  return depth(z)
end.

```

As shown in Figure 4, the elements  $\mathcal{E}$  of a model  $\mathcal{C}$  can be organized as a forest graph, where each tree is a case element module. We used the CMMN element icons to represent the nodes of the tree. Figure 4 shows a forest with two trees  $\lceil x \rceil$ ,  $\lceil z \rceil$ , and a subtree  $\lceil y \rceil$ . It also shows six leaves, namely  $d, e, a, b, c,$  and  $z$ .

Note that,

Length for Figure 1 is  $CL(\text{example1}) = 1$ .  
 Length for Figure 2 is  $CL(\text{example2}) = 2$ .  
 Length for Figure 3 is  $CL(\text{example3}) = 3$ .  
 Length for  $\lceil x \rceil$  in Figure 3 is  $CL(\lceil x \rceil) = 3$ .  
 Length for  $\lceil y \rceil$  in Figure 3 is  $CL(\lceil y \rceil) = 2$ .  
 Length for  $\lceil z \rceil$  in Figure 3 is  $CL(\lceil z \rceil) = 1$ .

DEFINITION 9. (*Complexity metric*): The complexity of a model  $\mathcal{C}$  denoted by  $CC(\mathcal{C})$  is defined as,

$$CC(\emptyset) = 0, \text{ otherwise } CC(\mathcal{C}) = \sum_{i \in \mathcal{E} \cup \mathcal{A}} W_i \quad (3.9)$$

Where, the weight,  $W_i$  is given in Table 3.

The complexity of a model resembles a cognitive complexity metric with cognitive weights. There are several cognitive complexity metrics defined for BPM [29, 10, 11, 7]. However, those cognitive complexity metrics are not applicable to CMMN, because they are based on control structures [9] like sequence, branching, iterations, etc., which are

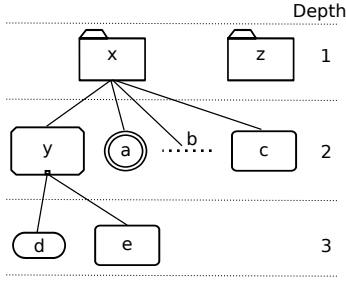


Figure 4: Tree view of elements in Example 3

not present in CMMN. For the CMMN complexity metric  $CC(\mathcal{C})$ , instead we assign weights to elements in  $\mathcal{E}$  and annotators in  $\mathcal{A}$ . The weights were assigned based on the authors' intuition. Higher weight were given to less frequently used elements and to annotators that increase the number of elements in the visual canvas, because less frequently used elements require better recall capabilities by users and clutter by annotators makes the model harder to read.

The complexity of a module  $\lceil z \rceil$  is defined as,

$$CC(\lceil z \rceil) = \sum_{i \in \mathcal{E}_z \cup \mathcal{A}_z} W_i$$

Note that,

Complexity for Figure 1 is  $CC(\text{example1}) = 1$ .

Complexity for Figure 2 is  $CC(\text{example2}) = 7$

(weights  $w=1, o=1, p=0, q=1, k=1, l=0, j=3$ ).

Complexity for Figure 3 is  $CC(\text{example3}) = 11$

(weights  $z=1, x=1, a=2, b=0, c=1, y=1, d=1, e=1, h=1, i=0, j=0, k=1, l=0, m=1$ ).

Complexity for  $\lceil x \rceil$  in Figure 3 is  $CC(\lceil x \rceil) = 10$

(weights  $x=1, a=2, b=0, c=1, y=1, d=1, e=1, h=1, i=0, j=0, k=1, l=0, m=1$ ).

Complexity for  $\lceil y \rceil$  in Figure 3 is  $CC(\lceil y \rceil) = 3$

(weights  $y=1, d=1, e=1$ ).

Complexity for  $\lceil z \rceil$  in Figure 3 is  $CC(\lceil z \rceil) = 1$ .

## 4. THEORETICAL VALIDATION

In this section a theoretical validation of each metric is conducted using the Briand *et al.* framework [3]. The complexity metric is further validated against the nine properties for software complexity measures defined by Weyuker [36].

### 4.1 Briand's framework

The Briand *et al.* framework [4] categorizes software metrics into size, length, complexity, cohesion, and coupling. The framework is based on systems and modules. Briand *et al.* defines a system  $S$  as a pair  $S = \langle E, R \rangle$ , where  $E$  represents a set of elements of  $S$ , and  $R$  is a binary relationship on  $E$  ( $R \subseteq E \times E$ ). A module of  $S$  is defined as  $m = \langle E_m, R_m \rangle$  for  $E_m \subseteq E$ ,  $R_m \subseteq E_m \times E_m$  and  $R_m \subseteq R$ .

For our purposes a system corresponds to a model  $\mathcal{C}$  and a module corresponds to a module  $\lceil m \rceil$ ,  $m \in \mathcal{M}$ .  $E$  corresponds to modeling elements  $\mathcal{E}$  ( $E = \mathcal{E}$ ) as described in Table 1.  $R$  corresponds to the two binary relationships in  $\mathcal{C}$  ( $\mathcal{U}$  and  $\mathcal{V}$ ). We define  $R$  as follows,

$$R = \{ \llbracket a, b \rrbracket \mid \llbracket a, b \rrbracket \in \mathcal{U} \} \cup \{ \langle a, b \rangle \mid \langle a, b \rangle \in \mathcal{V} \}$$

Note that  $\llbracket a, b \rrbracket \neq \langle a, b \rangle$ ,  $\llbracket a, b \rrbracket$  is an unordered pair, while  $\langle a, b \rangle$  is an ordered pair.

$\mathcal{E} \cup \mathcal{A}$	Description	Weight
	case element	1
	stage element	1
	discretionary stage element	2
	plan fragment element	3
	case file item element	1
	task element	1
	discretionary task element	2
	event listener element	2
	milestone element	1
	connector (sentry) element	0
	collapsed planing table	1
	expanded planing table	2
	auto complete	2
	collapsed	0
	expanded	1
	manual activation	1
	repetition	1
	required	1
	entry criterion with associated connector	1
	entry criterion without a connector	2
	exit criterion with associated connector	1
	exit criterion without a connector	3
	non-blocking human	1
	process	0
	case referring to a case element not in this model	0
	case referring to a case element in this model	1
	participant	0
	timer	0

Table 3: CMMN weights

We will use the terminology used in the Briand *et al.* framework [4] to introduce each property followed by a short proof against our metrics.

#### 4.1.1 Size

Briand *et al.* define a function  $Size(S)$  characterized by three properties.

**SIZE 1. Non-negativity.** The size of a model  $S = \langle E, R \rangle$  is non-negative.

$$(S = \langle E, R \rangle) \implies Size(S) \geq 0$$

**Proof:** By definition 7,  $Size$  is the cardinality of  $\mathcal{E}$  and the cardinality of a set cannot be negative.

$$\therefore CS(\mathcal{C}) = |\mathcal{E}| \geq 0 \quad \square$$

**SIZE 2. Null value.** The size of a model  $S = \langle E, R \rangle$  is zero if  $E$  is empty.

$$(S = \langle E, R \rangle \wedge E = \emptyset) \implies Size(S) = 0$$

**Proof:** By definition, the cardinality of an empty set is zero.

$$\therefore (\mathcal{E} = \emptyset) \implies CS(\mathcal{C}) = |\mathcal{E}| = |\emptyset| = 0 \quad \square$$

**SIZE 3. Module additivity.** The size of a module  $S = \langle E, R \rangle$  is equal to the sum of the sizes of two of its modules  $m_1 = \langle E_{m_1}, R_{m_1} \rangle$  and  $m_2 = \langle E_{m_2}, R_{m_2} \rangle$  such that any element of  $S$  is in either  $m_1$  or in  $m_2$ .

$$(m_1 \subseteq S \wedge m_2 \subseteq S \wedge E = E_{m_1} \cup E_{m_2} \wedge E_{m_1} \cap E_{m_2} = \emptyset) \implies Size(S) = Size(m_1) + Size(m_2)$$

**Proof:** Consider a model  $\mathcal{C}$  with two disjoint modules  $\lceil x \rceil$  and  $\lceil y \rceil$  such that each element in  $\mathcal{C}$  is either in  $\lceil x \rceil$  or in  $\lceil y \rceil$ , but not both ( $(\mathcal{E}_x \cap \mathcal{E}_y = \emptyset) \wedge (\mathcal{E} = \mathcal{E}_x \cup \mathcal{E}_y)$ ). It follows, based on (3.5), that  $x$  and  $y$  are the only two case elements in  $\mathcal{L}$  ( $\mathcal{L} = \{x, y\}$ ), thus  $\mathcal{E}$  is partitioned by  $\lceil x \rceil$  and  $\lceil y \rceil$ .

$$\therefore CS(\mathcal{C}) = |\mathcal{E}| = |\mathcal{E}_x| + |\mathcal{E}_y| = CS(\lceil x \rceil) + CS(\lceil y \rceil) \quad \square$$

#### 4.1.2 Length

Briand *et al.* define a function  $Length(S)$  characterized by five properties.

**LENGTH 1. Non-negativity.** The length of a model  $S = \langle E, R \rangle$  is non-negative.

$$(S = \langle E, R \rangle) \implies Length(S) \geq 0$$

**Proof:**  $CL$  is defined as the maximum nesting depth of a model, and calculated with algorithm  $CL(\mathcal{C})$ . Analyzing algorithm  $CL(\mathcal{C})$ , the variables ( $m$  and  $d$ ) are initialized to zero, and only increased by one or assigned the maximum of itself and depth which always returns  $d + 1$ .

$$\therefore CL(\mathcal{C}) \geq 0 \quad \square$$

**LENGTH 2. Null value.** The length of a model  $S = \langle E, R \rangle$  is zero if  $E$  is empty.

$$(S = \langle E, R \rangle \wedge E = \emptyset) \implies Length(S) = 0$$

**Proof:** Consider an empty model  $\mathcal{C}$ , then  $\mathcal{E} = \emptyset \implies \mathcal{L} = \emptyset$  (because  $\mathcal{L} \subseteq \mathcal{E}$ ). Analyzing algorithm  $CL(\mathcal{C})$ , it initializes  $m$  to zero, and if  $\mathcal{L} = \emptyset$  then ‘depth( $m, c$ )’ is never invoked, forcing the return of  $m$  which is zero.

$$\therefore (\mathcal{E} = \emptyset) \implies CL(\mathcal{C}) = 0 \quad \square$$

**LENGTH 3. Non-increasing monotonicity for connected components.** Adding relationships between elements of a module  $m$  does not increase the length of the model  $S = \langle E, R \rangle$ .

$$(S = \langle E, R \rangle \wedge m = \langle E_{m'}, R_m \rangle \wedge m \subseteq S \wedge m \text{ is a connected component of } S \wedge S' = \langle E, R' \rangle \wedge R' = R \cup \{\langle e_1, e_2 \rangle\} \wedge \langle e_1, e_2 \rangle \notin R \wedge e_1 \in E_{m'} \wedge e_2 \in E_{m'}) \implies Length(S) \geq Length(S')$$

**Proof:** Consider a model  $\mathcal{C}$  with two modeling elements in a module  $(a, b \in \lceil x \rceil)$ . There are two cases,

**case 1**  $a$  and  $b$  are in different scope. They cannot be related by scope, as that will require moving them within submodules of module  $\lceil x \rceil$  adding  $(a, b)$  to  $R$  which will change the structure of modules violating  $\langle e_1, e_2 \rangle \notin R$ . They cannot be related by an event, because by definition 6 to be related by an event they are necessarily in the same scope.

**case 2**  $a$  and  $b$  are in the same scope. They are already related by scope  $\llbracket a, b \rrbracket \in \mathcal{U}$ . Assuming  $\langle a, b \rangle \notin \mathcal{V}$ , we can relate them by an event and add them to  $\mathcal{V}_m$ , which corresponds to adding  $\langle a, b \rangle$  to  $R'$  and leaving  $R$  invariant.

Therefore,  $a$  and  $b$  can be related only by an event, and the scope relationship  $\mathcal{U}$  is not affected. Then, neither the length of  $\lceil x \rceil$  ( $CL(\lceil x \rceil)$ ), nor the length of the model  $\mathcal{C}$  ( $CL(\mathcal{C})$ ) has changed.

The maximum nesting depth of a forest is greater than or equal to the maximum nesting depth of any of the trees or subtrees, and  $\lceil x \rceil$  is either a tree or a subtree on model  $\mathcal{C}$ .

$$\therefore CL(\mathcal{C}) \geq CL(\lceil x \rceil) \quad \square$$

**LENGTH 4. Non-decreasing monotonicity for non-connected components.** Adding relationships between the elements of two modules  $m_1$  and  $m_2$  does not decrease the length of the model  $S = \langle E, R \rangle$ .

$$(S = \langle E, R \rangle \wedge m_1 = \langle E_{m_1}, R_{m_1} \rangle \wedge m_2 = \langle E_{m_2}, R_{m_2} \rangle \wedge m_1 \subseteq S \wedge m_2 \subseteq S \wedge m_1, m_2 \text{ are separate connected components of } S \wedge S' = \langle E, R' \rangle \wedge R' = R \cup \{\langle e_1, e_2 \rangle\} \wedge \langle e_1, e_2 \rangle \notin R \wedge e_1 \in E_{m_1} \wedge e_2 \in E_{m_2}) \implies Length(S) \geq Length(S')$$

**Proof:** Adding relationships between elements of two modules is not allowed in CMMN models.  $\square$

**LENGTH 5. Disjoint modules.** The length of a model  $S = \langle E, R \rangle$  made up of two disjoint modules  $m_1$  and  $m_2$  is equal to the maximum of the lengths of the modules  $m_1$  and  $m_2$ .

$$(S = m_1 \cup m_2 \wedge m_1 \cap m_2 = \emptyset \wedge E = E_{m_1} \cup E_{m_2}) \implies Length(S) = \max(Length(m_1), Length(m_2))$$

**Proof:** Consider a model  $\mathcal{C}$  with two disjoint modules  $\lceil x \rceil$  and  $\lceil y \rceil$ , such that  $\mathcal{E} = \mathcal{E}_x \cup \mathcal{E}_y \wedge \mathcal{E}_x \cap \mathcal{E}_y = \emptyset$ . Therefore, by (3.5)  $x$  and  $y$  are the only two case elements in  $\mathcal{L}$  ( $\mathcal{L} = \{x, y\}$ ) and they create the only two trees ( $\lceil x \rceil, \lceil y \rceil$ ) in the forest. The maximum nesting depth of a forest is equal to the maximum nesting depth of its trees.

$$\therefore CL(\mathcal{C}) = \max(CL(\lceil x \rceil), CL(\lceil y \rceil)) \quad \square$$

### 4.1.3 Complexity

Briand *et al.* define a function  $Complexity(S)$  characterized by five properties. Complexity for Briand *et al.* is distinct from cognitive complexity, as they state that complexity in the framework is an intrinsic attribute of an object and not a perceived psychological complexity.

COMPLEXITY 1. *Non-negativity.* The complexity of a model  $S = \langle E, R \rangle$  must be non-negative.

$$(S = \langle E, R \rangle) \implies Complexity(S) \geq 0$$

**Proof:** By definition 9,  $CC(\mathcal{C}) = \sum_{i \in \mathcal{E} \cup \mathcal{A}} W_i$ , where weight  $W_i$  is a positive integer from 0 to 3 given by Table 3. Suppose we replace each element in  $\mathcal{E}$  for its weight and call the resulting set  $\mathcal{E}^W$ , and each annotator in  $\mathcal{A}$  for its weight and call that set  $\mathcal{A}^W$ . Then,  $CC(\mathcal{C}) = \sum_{i \in \mathcal{E} \cup \mathcal{A}} W_i = \sum_{p \in \mathcal{E}^W \cup \mathcal{A}^W} p \wedge p \in \mathbb{N}_0$ , therefore  $(\sum_{p \in \mathcal{E}^W \cup \mathcal{A}^W} p) \in \mathbb{N}_0$ .

$$\therefore CC(\mathcal{C}) \geq 0. \quad \square$$

COMPLEXITY 2. *Null value.* The complexity of a model  $S = \langle E, R \rangle$  is zero if  $R$  is empty.

$$(S = \langle E, R \rangle \wedge R = \emptyset) \implies Complexity(S) = 0$$

**Proof:** By definition 9, the complexity of an empty model is zero.

$$\therefore (\mathcal{C} = \emptyset) \implies R = \mathcal{U} = \mathcal{V} = \emptyset \wedge CC(\mathcal{C}) = 0 \quad \square$$

COMPLEXITY 3. *Symmetry.* The complexity of a model  $S = \langle E, R \rangle$  does not depend on the convention chosen to represent the relationships between its elements.

$$(S = \langle E, R \rangle \wedge S^{-1} = \langle E, R^{-1} \rangle) \implies Complexity(S) = Complexity(S^{-1})$$

**Proof:** For a model  $\mathcal{C}$ ,  $R$  is given by the two relationships  $\mathcal{U}$  and  $\mathcal{V}$ . By definition 9,  $CC(\mathcal{C}) = \sum_{i \in \mathcal{E} \cup \mathcal{A}} W_i$ , which does not depend on  $\mathcal{U}$  or  $\mathcal{V}$ , or the convention used to represent  $\mathcal{U}$  and  $\mathcal{V}$ .

$$\therefore CC(\mathcal{C}) = CC(\mathcal{C}') \quad \square$$

COMPLEXITY 4. *Module monotonicity.* The complexity of a model  $S = \langle E, R \rangle$  is no less than the sum of the complexities of any two of its modules with no relationships in common.

$$(S = \langle E, R \rangle \wedge m_1 = \langle E_{m_1}, R_{m_1} \rangle \wedge m_2 = \langle E_{m_2}, R_{m_2} \rangle \wedge m_1 \cup m_2 \subseteq S \wedge R_{m_1} \cap R_{m_2} = \emptyset) \implies Complexity(S) \geq Complexity(m_1) + Complexity(m_2)$$

**Proof:** Consider a model  $\mathcal{C}$  with two modules  $\lceil m_1 \rceil$  and  $\lceil m_2 \rceil$  such that  $\mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \subseteq \mathcal{E} \wedge \mathcal{U}_{m_1} \cap \mathcal{U}_{m_2} = \emptyset \wedge \mathcal{V}_{m_1} \cap \mathcal{V}_{m_2} = \emptyset$ . We can ignore relationship  $R = \mathcal{U} \cup \mathcal{V}$ , because by definition 9,  $CC(\mathcal{C})$  does not depend on  $\mathcal{U}$  or  $\mathcal{V}$ .

Because  $\mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \subseteq \mathcal{E}$ , we can define  $T$  such that  $\mathcal{E} = T \cup \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \wedge T \cap \mathcal{E}_{m_1} = T \cap \mathcal{E}_{m_2} = T \cap \mathcal{E}_{m_1} \cap \mathcal{E}_{m_2} = \emptyset$ .

Annotators in  $\mathcal{A}$  are associated with elements in  $\mathcal{E}$ , therefore we can define a function  $f$  such that  $f : \mathcal{A} \rightarrow \mathcal{E}$ . We can separate the elements of  $\mathcal{A}$  such that  $\mathcal{A}_{m_1} = \{x \mid f(x) \in \mathcal{E}_{m_1}\}$ ,  $\mathcal{A}_{m_2} = \{x \mid f(x) \in \mathcal{E}_{m_2}\}$ , and  $\mathcal{A}_T = \{x \mid f(x) \in T\}$ .

By way of contradiction, assume  $\mathcal{A}_{m_1} \cap \mathcal{A}_{m_2} \cap \mathcal{A}_T = \{a\}$ . That means there is an annotator 'a' in  $\mathcal{A}$  for which  $f(a) \notin T \cup \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2}$ , but  $\mathcal{E} = T \cup \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2}$ . We have an annotator 'a' without an image in  $\mathcal{E}$  which contradicts our definition of  $f$ . Therefore, 'a' cannot exist, and  $\mathcal{A} = \mathcal{A}_{m_1} \cup \mathcal{A}_{m_2} \cup \mathcal{A}_T \wedge \mathcal{A}_{m_1} \cap \mathcal{A}_{m_2} = \mathcal{A}_{m_1} \cap \mathcal{A}_T = \mathcal{A}_{m_2} \cap \mathcal{A}_T = \mathcal{A}_{m_1} \cap \mathcal{A}_{m_2} \cap \mathcal{A}_T = \emptyset$ .

We have two cases for  $T$ ,

**case 1**  $T = \emptyset$ , in which case  $\mathcal{E} = \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \wedge \mathcal{E}_{m_1} \cap \mathcal{E}_{m_2} = \emptyset$ . By (3.5) and complexity 5, we have that  $CC(\mathcal{C}) = CC(\lceil m_1 \rceil) + CC(\lceil m_2 \rceil)$

**case 2**  $T \neq \emptyset$ , in which case  $T \subseteq \mathcal{E} \wedge \mathcal{E}_{m_1} \cap \mathcal{E}_{m_2} = T \cap \mathcal{E}_{m_1} = T \cap \mathcal{E}_{m_2} = T \cap \mathcal{E}_{m_1} \cap \mathcal{E}_{m_2} = \emptyset$ . Thus,  $CC(\mathcal{C}) = \sum_{i \in \mathcal{E} \cup \mathcal{A}} W_i = \sum_{i \in \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \cup T \cup \mathcal{A}_{m_1} \cup \mathcal{A}_{m_2} \cup \mathcal{A}_T} W_i = \sum_{i \in \mathcal{E}_{m_1} \cup \mathcal{A}_{m_1}} W_i + \sum_{i \in \mathcal{E}_{m_2} \cup \mathcal{A}_{m_2}} W_i + \sum_{i \in T \cup \mathcal{A}_T} W_i = CC(\lceil m_1 \rceil) + CC(\lceil m_2 \rceil) + CC(T)$

$$\therefore CC(\mathcal{C}) \geq CC(\lceil m_1 \rceil) + CC(\lceil m_2 \rceil) \quad \square$$

COMPLEXITY 5. *Disjoint module additivity.* The complexity of a model  $S = \langle E, R \rangle$  composed of two disjoint modules  $m_1$  and  $m_2$  is equal to the sum of the complexities of the two modules.

$$(S = \langle E, R \rangle \wedge S = m_1 \cup m_2 \wedge m_1 \cap m_2 = \emptyset) \implies Complexity(S) = Complexity(m_1) + Complexity(m_2)$$

**Proof:** Consider a model  $\mathcal{C}$  with two modules  $\lceil m_1 \rceil$  and  $\lceil m_2 \rceil$  such that  $\mathcal{E} = \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \wedge \mathcal{E}_{m_1} \cap \mathcal{E}_{m_2} = \emptyset$ . By (3.5),  $m_1$  and  $m_2$  are case elements and  $\mathcal{L} = \{m_1, m_2\}$ .

Annotators in  $\mathcal{A}$  are associated with elements in  $\mathcal{E}$ . Therefore, we can define a function  $f$  such that  $f : \mathcal{A} \rightarrow \mathcal{E}$ . We can separate the elements of  $\mathcal{A}$  such that  $\mathcal{A}_{m_1} = \{x \mid f(x) \in \mathcal{E}_{m_1}\}$  and  $\mathcal{A}_{m_2} = \{x \mid f(x) \in \mathcal{E}_{m_2}\}$ .

By way of contradiction, assume  $\mathcal{A}_{m_1} \cap \mathcal{A}_{m_2} = \{a\}$ . That means there is an annotator 'a' in  $\mathcal{A}$  for which  $f(a) \notin \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2}$ , but  $\mathcal{E} = \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2}$ . We have an annotator 'a' without an image in  $\mathcal{E}$  which contradicts our definition of  $f$ . Therefore, 'a' cannot exist, and  $\mathcal{A} = \mathcal{A}_{m_1} \cup \mathcal{A}_{m_2} \wedge \mathcal{A}_{m_1} \cap \mathcal{A}_{m_2} = \emptyset$ .

Now, we have that  $CC(\mathcal{C}) = \sum_{i \in \mathcal{E} \cup \mathcal{A}} W_i = \sum_{i \in \mathcal{E}_{m_1} \cup \mathcal{E}_{m_2} \cup \mathcal{A}_{m_1} \cup \mathcal{A}_{m_2}} W_i = \sum_{i \in \mathcal{E}_{m_1} \cup \mathcal{A}_{m_1}} W_i + \sum_{i \in \mathcal{E}_{m_2} \cup \mathcal{A}_{m_2}} W_i = CC(\lceil m_1 \rceil) + CC(\lceil m_2 \rceil)$ .

$$\therefore CC(\mathcal{C}) = CC(\lceil m_1 \rceil) + CC(\lceil m_2 \rceil) \quad \square$$

## 4.2 Weyuker's properties

Weyuker [36] proposed a set of nine properties of complexity software metrics that have been widely used to validate business process metrics [23]. Briand *et al.* [4] found that the Weyuker [36] properties are consistent with complexity in his framework.

The Weyuker [36] notation uses  $P$ ,  $Q$ , and  $R$  to denote programs, which for our purposes corresponds to CMMN models. To avoid confusion with the Briand *et al.* [4] framework, we will use  $T$  instead of  $R$ . Weyuker uses the set cardinality operator  $|P|$  to denote complexity, which we already used to denote cardinality. Therefore, we will use  $\|P\|$  as the complexity of model  $P$ . Weyuker expects that for any  $P$ , its complexity  $\|P\|$  is a nonnegative number, hence complexity can be compared and ordered.

$$\|P\| \leq \|Q\| \vee \|Q\| \leq \|P\|$$

We will use the terminology used by Weyuker [36] and the classification by Srinivasan and Devi [30] to introduce each property, followed by a short proof against our complexity metric  $CC$ .

PROPERTY 1. *Non-coarseness.* A metric should not rank all models as equally complex.

$$(\exists P)(\exists Q)(\|P\| \neq \|Q\|)$$

**Proof:** Let  $P$  be example 2 (see Figure 2) with  $CC(P) = 7$ , and  $Q$  be example 3 (see Figure 3) with  $CC(Q) = 11$ .

$\therefore (\exists P)(\exists Q)(CC(P) \neq CC(Q)) \quad \square$

**PROPERTY 2. Granularity.** A metric should rank only a finite number of models with the same complexity.

Let  $c$  be a non-negative number, then there are only finitely many models of complexity  $c$ .

**Proof:** Assuming a model can only be renamed (see property 8) in a finite number of ways, then consider a number  $c \in \mathbb{N}_0$ , and a model  $P$  such that  $CC(P) = c$ . By definition 9,  $CC(P)$  is calculated using  $\mathcal{E}_P$  and  $\mathcal{A}_P$ . We need to show there is a finite number of  $\mathcal{E}_P \cup \mathcal{A}_P$  sets such that  $CC(P) = c$ . Note that if  $c = 0$  then  $\mathcal{E}_P \cup \mathcal{A}_P = \emptyset \wedge CC(\emptyset)$ , thus there is only one  $\mathcal{E}_P \cup \mathcal{A}_P$  that gives  $c = 0 \quad \square$ .

We start by proving that  $\mathcal{E}_P \cup \mathcal{A}_P$  is finite for  $c > 0$ . The only element in  $\mathcal{E}_P$  (see Table 1) that has a weight of 0 is the connector (see Table 3), but a connector must be associated with a sentry in  $\mathcal{A}_P$  with weight of 1 (entry or exit criteria). We cannot add an infinite number of connectors to  $\mathcal{E}_P$  without adding sentries and changing the value of  $CC(P)$ . Thus,  $CC(P) = c \implies |\mathcal{E}_P| \neq \infty$ , and model  $P$  has a size  $CS(P) = |\mathcal{E}_P| = n$ . Set  $\mathcal{E}_P$  is finite ( $(\exists n)(|\mathcal{E}_P| \leq n)$ ). By definition 1, each element in  $\mathcal{A}_P$  is associated with a single element in  $\mathcal{E}_P$ . In  $\mathcal{A}_P$  (see Table 2) markers and decorators are bound to  $|\mathcal{E}_P|$ , but sentries are not. However, sentries (entry and exit criterion) weights range from 1 to 3. We cannot have an infinite numbers of sentries, because the number is bound by  $c$  (i.e.  $\sum_{i \in \{x|x \in \{1,2,3\} \times |\mathcal{A}_P|\}} i \leq c$ ). Therefore,  $\mathcal{A}_P$  is also finite and  $(\exists m)(|\mathcal{A}_P| \leq m)$ . Moreover  $\mathcal{E}_P \cap \mathcal{A}_P = \emptyset \implies |\mathcal{E}_P| + |\mathcal{A}_P| = |\mathcal{E}_P \cup \mathcal{A}_P| \leq n + m$ . Therefore,  $\mathcal{E}_P \cup \mathcal{A}_P$  is a finite set.

Without loss of generality, we can summarize Table 3 into four weight categories (0, 1, 2, 3), and we know  $|\mathcal{E}_P \cup \mathcal{A}_P| \leq n + m$ . Using brute force, we can count all combinations with repetition of four categories (0, 1, 2, 3) into 1 to  $n + m$  slots, using  $C(n + r - 1, r)$ . That will give us all possible sets  $\mathcal{E}_P \cup \mathcal{A}_P$  that can produce, among other complexities,  $CC(P) = c$ . We calculate  $\sum_{r=1}^{n+m} C(4+r-1, r) = 2^{4+n+m-1}$ . Therefore, there is a finite number of sets  $\mathcal{E}_P \cup \mathcal{A}_P$  such that  $CC(P) = \sum_{i \in \mathcal{E}_P \cup \mathcal{A}_P} W_i = c. \quad \square$

**PROPERTY 3. Non-uniqueness (Notion of equivalence).** A metric should allow some models to have the same complexity.

$(\exists P)(\exists Q)(P \neq Q \wedge \|P\| = \|Q\|)$

**Proof:** Let  $P$  be example 2 (see Figure 2) with  $CC(P) = 7$ , and  $Q$  a similar model, but changing modeling element  $o$  (case file item) to a task  $t$ . Thus,  $Q$  is different from  $P$  by one modeling element. The complexity of  $Q$ , namely,  $CC(Q) = 7$ , because the weight for a task is the same as the weight for a case file item (see Table 3).

$\therefore (\exists P)(\exists Q)(P \neq Q \wedge CC(P) = CC(Q)) \quad \square$

**PROPERTY 4. Design details are important.** Two distinct but equivalent models that compute the same function need not have the same complexity.

$(\exists P)(\exists Q)(P \equiv Q \wedge \|P\| \neq \|Q\|)$ .

**Proof:** Let  $P$  be example 3 (see Figure 3) with  $CC(P) = 11$ , and  $Q$  a similar model, but includes an extra task  $t$

inside case  $x$ . Task  $t$  is a dummy task that does nothing when it executes (a ‘skip’ statement). Operationally,  $Q$  is equivalent to  $P$  ( $Q \equiv P$ ), because they compute the same function. However, the complexity of  $Q$  is  $CC(Q) = 12$ , because task  $t$  adds a weight of 1 (see Table 3).

$\therefore (\exists P)(\exists Q)(Q \equiv P \wedge CC(P) \neq CC(Q)) \quad \square$

**PROPERTY 5. Monotonicity.** The complexity of two models joined together is greater than or equal to the complexity of either model considered separate.

$(\forall P)(\forall Q)(\|P\| \leq \|Q; P\| \wedge \|Q\| \leq \|P; Q\|)$ .

**Proof:** Let  $CC(P) = \sum_{i \in \mathcal{E}_P \cup \mathcal{A}_P} W_i$ , and  $CC(Q) = \sum_{i \in \mathcal{E}_Q \cup \mathcal{A}_Q} W_i$ . Then,  $CC(P; Q) = \sum_{i \in \mathcal{E}_P \cup \mathcal{E}_Q \cup \mathcal{A}_P \cup \mathcal{A}_Q} W_i = \sum_{i \in \mathcal{E}_Q \cup \mathcal{E}_P \cup \mathcal{A}_Q \cup \mathcal{A}_P} W_i = CC(Q; P)$ . We have two cases,

**case 1**  $P = \emptyset \vee Q = \emptyset$ . Assume  $P$  is an empty model, then  $\mathcal{E}_P = \mathcal{A}_P = \emptyset$ . Therefore,  $CC(P; Q) = \sum_{i \in \emptyset \cup \mathcal{E}_Q \cup \emptyset \cup \mathcal{A}_Q} W_i = \sum_{i \in \mathcal{E}_Q \cup \mathcal{A}_Q} W_i = CC(Q)$ . Assuming  $Q$  is an empty model gives the same result  $CC(P; Q) = CC(P)$ . Assuming both  $P$  and  $Q$  are empty leads to the same result  $CC(P; Q) = CC(P) = CC(Q) = 0$ .

**case 2**  $P \neq \emptyset \wedge Q \neq \emptyset$ . Then,  $CC(P; Q) = \sum_{i \in \mathcal{E}_P \cup \mathcal{E}_Q \cup \mathcal{A}_P \cup \mathcal{A}_Q} W_i = \sum_{i \in \mathcal{E}_P \cup \mathcal{A}_P} W_i + \sum_{i \in \mathcal{E}_Q \cup \mathcal{A}_Q} W_i = CC(P) + CC(Q)$ .

$\therefore (\forall P)(\forall Q)(CC(P) \leq CC(P) + CC(Q) \wedge CC(Q) \leq CC(P) + CC(Q)) \quad \square$

**PROPERTY 6. Nonequivalence of interaction.** Two models with the same complexity when each is joined to a third model the resulting complexity may be different between the two.

a:  $(\exists P)(\exists Q)(\exists T)(\|P\| = \|Q\| \wedge \|P; T\| \neq \|Q; T\|)$   
b:  $(\exists P)(\exists Q)(\exists T)(\|P\| = \|Q\| \wedge \|T; P\| \neq \|T; Q\|)$

**Proof:** For complexity  $CC$  the order of concatenation of models is not important. Let  $CC(P) = \sum_{i \in \mathcal{E}_P \cup \mathcal{A}_P} W_i$ , and  $CC(T) = \sum_{i \in \mathcal{E}_T \cup \mathcal{A}_T} W_i$ . Then,  $CC(P; T) = \sum_{i \in \mathcal{E}_P \cup \mathcal{E}_T \cup \mathcal{A}_P \cup \mathcal{A}_T} W_i = \sum_{i \in \mathcal{E}_T \cup \mathcal{E}_P \cup \mathcal{A}_T \cup \mathcal{A}_P} W_i = CC(T; P)$ . Therefore,  $CC(P; T) = CC(T; P)$  which is the same as  $\|P; T\| = \|T; P\|$ , and we have only one case.

Let  $P$  be example 2 (see Figure 2) with  $CC(P) = 7$ , and  $Q$  similar to  $P$  but changing modeling element  $o$  to a task  $t$ . Thus,  $Q$  is different from  $P$  by one modeling element, still  $CC(P) = CC(Q) = 7$  (same as in property 3). Now let the case annotator ‘l’ in  $P$  invoke case  $z$ , and case annotator ‘l’ in  $Q$  invoke case  $x$ . That does not change  $CC(P)$  or  $CC(Q)$ , because neither case  $z$  or case  $x$  are in the models, therefore the weight of ‘l’ remains 0 (see Table 3).

Let  $T$  be example 1 (see Figure 1) with  $CC(T) = 1$ . Concatenating  $P$  with  $T$  produces  $CC(P; T) = 9$  (weights  $z=1, w=1, o=1, p=0, q=1, k=1, l=1, j=3$ ), where ‘l’=1, because case  $z$  is in  $P; T$ . However, concatenating  $Q$  with  $T$  is  $CC(Q; T) = 8$  (weights  $z=1, w=1, o=1, p=0, q=1, k=1, l=0, j=3$ ), where ‘l’=0, because case  $x$  is not in  $Q; T$ .

$\therefore (\exists P)(\exists Q)(\exists T)(CC(P) = CC(Q) \wedge CC(P; T) \neq CC(Q; T)) \quad \square$



PROPERTY 7. *Permutation.* Complexity should be responsive to the order of statements.

$$(\exists P)(\exists Q)(\text{Permutation}(Q, P) \wedge \|P\| \neq \|Q\|)$$

**Proof:** Let  $P$  be example 2 (see Figure 2) with  $CC(P) = 7$ , and  $Q$  be similar to  $P$  but with connector  $p$  attached to exit criteria ‘j’ instead of entry criteria ‘k’. Thus,  $Q$  is a permutation of  $P$  with  $CC(Q) = 6$ , because in  $CC(Q)$  the weight of ‘k’ increased from 1 to 2, and the weight of ‘j’ decreased from 3 to 1 (see Table 3).

$$\therefore (\exists P)(\exists Q)(\text{Permutation}(Q, P) \wedge CC(P) \neq CC(Q)) \quad \square$$

PROPERTY 8. *Renaming.* Complexity should not be affected by renaming.

$$(\forall P)(\forall Q)(\text{Rename}(Q, P) \wedge \|P\| = \|Q\|)$$

**Proof:** The names of elements and annotators do not affect  $CC$ . Let  $P$  be example 2 (see Figure 2) with  $CC(P) = 7$ , and  $Q$  be similar to  $P$  but with different names. Suppose we rename  $w, o, p, q, k, l, j$  in  $\mathcal{E}_P$  to  $a, b, c, d, e, f, q$  in  $\mathcal{E}_Q$ . Thus,  $Q$  is a renaming of  $P$ . Note that  $Q$  has the same number and type of modeling elements and annotators as  $P$ , and they are organized in exactly the same way. Thus, the complexity of  $Q$  remains  $CC(Q) = 7$ . This can be done with any model.

$$\therefore (\forall P)(\forall Q)(\text{Rename}(Q, P) \wedge CC(P) = CC(Q)) \quad \square$$

PROPERTY 9. *Interaction may increase complexity.*

$$(\exists P)(\exists Q)(\|P\| + \|Q\| < \|P; Q\|).$$

**Proof:** Let  $P$  be example 1 (see Figure 1) with  $CC(P) = 1$ , and  $Q$  be example 2 (see Figure 2) with  $CC(Q) = 7$ . Assume that case annotator ‘l’ in  $Q$  invokes case  $z$ , then in  $Q$  the weight of case annotator ‘l’ is 0, but when joined with  $P$  that has case  $z$ , the weight of annotator ‘l’ becomes 1, and  $CC(P; Q) = 9$  (see Table 3).

$$\therefore (\exists P)(\exists Q)(CC(P) + CC(Q) < CC(P; Q)) \quad \square$$

## 5. FINDINGS AND FUTURE RESEARCH

The three proposed metrics comply with the formal framework for software measurements defined by Briand *et al.* [4], and the proposed complexity metric also complies with the properties described by Weyuker [36]. However, both Briand *et al.*, and Weyuker assume that software systems are build using a procedural style, based on directed acyclic graphs. Briand *et al.* use directed acyclic graphs to describe their framework and to provide examples. Weyuker uses a procedural language to illustrate her properties. Therefore, work is required to understand the applicability of Briand *et al.*, and Weyuker to declarative systems.

The validation of complexity metrics for business processes requires both theoretical and empirical validation [23, 22, 30]. Work is required to conduct the empirical validation for the proposed metrics. Further work on empirical validation using suggestions from Misra *et al.* [22] and Kitchenham *et al.* [16] should be conducted.

The formalization of the CMMN model was sufficient to define and validate the metrics. But, CMMN claims an approach based on business artifacts [19], therefore further work is required to compare the formal CMMN model described here with a formalization of business artifacts.

Further research is required on the complexity metric. The weights given for the complexity metric  $CC(C)$  were

assigned based on the intuition of the authors, and further empirical work is needed to fine tune the weights.

The CMMN specification defines non-visual entities that are not represented in the CMMN models (they are not described in Tables 1 or 2). Some of those entities are roles and non-visualized system events. Empirical work is needed to understand the influence of CMMN non-visual entities on the complexity metric.

## 6. CONCLUSIONS

This study provides a formal description of CMMN [25] and three metrics have been defined. The metrics were successfully validated using the formal framework for software measurements defined by Briand *et al.* [4], and the properties for software complexity measures defined by Weyuker [36]. However, it is clear that both Briand *et al.* and Weyuker’s properties are designed for procedural models and not for declarative models. Therefore, further theoretical validation is required. In addition, further research is required to empirically validate the metrics and the complexity weights.

## 7. REFERENCES

- [1] W. Bandara and M. Rosemann. What Are the Secrets of Successful Process Modelling? Insights From an Australian Case Study. *Journal of Strategic Information Systems*, 10(3):47–68, 2005.
- [2] J. D. Berkley and R. G. Eccles. Rethinking the Corporate Workplace: Case Managers at Mutual Benefit Life. Case N9-492-015, Harvard Business School, Boston, MA, 1991.
- [3] L. C. Briand, K. E. Emam, and S. Morasca. Theoretical and empirical validation of software product measures. Technical report, International Software Engineering Research Network, 1995.
- [4] L. C. Briand, S. Morasca, and V. R. Basili. Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22(1):68–86, Jan. 1996.
- [5] J. Cardoso. Process control-flow complexity metric: An empirical validation. In *IEEE International Conference on Services Computing*, pages 167–173. IEEE, 2006.
- [6] J. Cardoso. Complexity analysis of BPEL web processes. *Software Process: Improvement and Practice*, 12:35–49, 2007.
- [7] E. oşkun. A New Complexity Metric for Business Process Models. Master, Atilim University, 2014.
- [8] T. Davenport and N. Nohria. Case Management and the Integration of Labor. *MIT Sloan Management Review*, 35(2):11–23, 1994.
- [9] K. Figl, J. Mendling, M. Strembeck, and J. C. Recker. On the cognitive effectiveness of routing symbols in process modeling languages. In W. Abramowicz and R. Tolksdorf, editors, *Proceedings of the 13th International Conference on Business Information Systems*, pages 230–241, Berlin, 2010. Springer.
- [10] V. Gruhn and R. Laue. Adopting the Cognitive Complexity Measure for Business Process Models. In *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on*, volume 1, pages 236–241, July 2006.

- [11] V. Gruhn and R. Laue. Complexity Metrics for Business Process Models. In W. Abramowicz and H. C. Mayer, editors, *9th international conference on business information systems*, pages 1–12, Klagenfurt, Austria, 2006.
- [12] V. Gruhn and R. Laue. Reducing the Cognitive Complexity of Business Process Models. In *8th IEEE International Conference on Cognitive Informatics*, pages 339–345, Kowloon, Hong Kong, 2009. IEEE.
- [13] M. Hauder, R. Kazman, and F. Matthes. Empowering End-Users to Collaboratively Structure Processes for Knowledge Work. In *Proceedings of the 18th International Conference on Business Information Systems (BIS)*, 2015.
- [14] J. B. Hill. The Case for Case Management Solutions. Gartner report G00235833, Gartner, June 2012.
- [15] R. Hull, J. Su, and R. Vaculin. Data Management Perspectives on Business Process Management: Tutorial Overview. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 943–948, New York, NY, USA, 2013. ACM.
- [16] B. Kitchenham, S. Pflieger, and N. Fenton. Towards a framework for software measurement validation. *Software Engineering, IEEE Transactions on*, 21(12):929–944, Dec 1995.
- [17] M. Kurz, W. Schmidt, A. Fleischmann, and M. Lederer. Leveraging CMMN for ACM: Examining the Applicability of a New OMG Standard for Adaptive Case Management. In *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management*, S-BPM ONE '15, pages 4:1–4:9, New York, NY, USA, 2015. ACM.
- [18] K. B. Lassen and W. M. P. V. D. Aalst. Complexity metrics for Workflow nets. *Information and Software Technology*, 51(3):610–626, Mar. 2009.
- [19] M. Marin, R. Hull, and R. Vaculín. Data Centric BPM and the Emerging Case Management Standard: A Short Survey. In M. La Rosa and P. Soffer, editors, *Business Process Management Workshops*, volume 132 of *Lecture Notes in Business Information Processing*, pages 24–30. Springer Berlin Heidelberg, 2013.
- [20] M. A. Marin, H. Lotriet, and J. A. Van Der Poll. Measuring Method Complexity of the Case Management Modeling and Notation (CMMN). In *Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology*, SAICSIT '14, pages 209:209–209:216, New York, NY, USA, 2014. ACM.
- [21] J. Melcher, J. Mendling, H. A. Reijers, and D. Seese. On Measuring the Understandability of Process Models (Experimental Results). In *1st Workshop on Empirical Research in BPM*, Ulm, Germany, 2009.
- [22] S. Misra, I. Akman, and R. Colomo-Palacios. Framework for evaluation and validation of software complexity measures. *Software, IET*, 6(4):323–334, August 2012.
- [23] G. M. Muketha, A. A. A. Ghani, M. H. Selamat, and R. Atan. A survey of business process complexity metrics. *Information Technology Journal*, 9(7):1336 – 1344, 2010.
- [24] OMG. Case Management Process Modeling (CMPM) Request for Proposal. Request for proposal, OMG, Needham, MA, 2009. Document bmi/09-09-23.
- [25] OMG. Case Management Model and Notation, version 1.0. Technical Report May, OMG, May 2014. Document formal/2014-05-05.
- [26] J. C. Recker, M. zur Muehlen, K. Siau, J. Erickson, and M. Indulska. Measuring method complexity: UML versus BPMN. In *15th Americas Conference on Information Systems*, pages 6–9, San Francisco, 2009.
- [27] H. A. Reijers, J. Rigter, and W. M. P. Van Der Aalst. The Case Handling Case. *International Journal of Cooperative Information Systems*, 12(3):365–391, 2003.
- [28] S. Schönig, M. Zeising, and S. Jablonski. Supporting collaborative work by learning process models and patterns from cases. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pages 60–69. IEEE, 2013.
- [29] J. Shao and Y. Wang. A new measure of software complexity based on cognitive weights. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 2, pages 1333 – 1338 vol.2, May 2003.
- [30] K. Srinivasan and T. Devi. Software Metrics Validation Methodologies in Software Engineering. *International Journal of Software Engineering & Applications (IJSEA)*, 5(6):87–102, November 2014.
- [31] B. R. Swan. *The Effects of Business Process Management Cognitive Resources and Individual Cognitive Differences on Outcomes of User Comprehension*. Phd, Virginia Polytechnic Institute and State University, 2007.
- [32] K. Swenson. *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done*. Landmark books. Meghan-Kiffer Press, 2010.
- [33] W. M. P. Van Der Aalst and P. J. S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In *Proceedings of the 2001 International ACM SIGGROUP*, pages 42–51, New York, 2001. ACM Press.
- [34] W. M. P. Van Der Aalst, M. Weske, and D. Grunbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.
- [35] I. Vanderfeesten, H. A. Reijers, J. Mendling, W. M. P. V. D. Aalst, and J. Cardoso. On a Quest for Good Process Models: The Cross-Connectivity Metric. In *The 20th International Conference on Advanced Information Systems Engineering*, pages 480–494, Berlin, 2008. Springer-Verlag.
- [36] E. Weyuker. Evaluating Software Complexity Measures. *IEEE Transactions on Software Engineering*, 14(9):1357–1365, 1988.